Ciências ULisboa | Informática

di.ciencias.ulisboa.pt

/difcul

Segurança e Confiabilidade

2024/2025

Trabalho 1 – 1ª fase

1 Objetivos

A parte prática da disciplina de Segurança e Confiabilidade pretende familiarizar os alunos com alguns dos problemas envolvidos na programação de aplicações distribuídas seguras, nomeadamente a gestão de chaves criptográficas, a geração de sínteses seguras, atestação remota, cifras e assinaturas digitais, e a utilização de canais seguros. O primeiro trabalho a desenvolver na disciplina será realizado utilizando a linguagem de programação Java e a API de segurança do Java, e é composto por duas fases. A primeira fase do projeto está enunciada neste documento, e tem como objetivo fundamental a construção de uma aplicação distribuída, focando-se essencialmente nas funcionalidades da aplicação. Na fase 2 do projeto, iremos dedicar os nossos esforços para garantir que todos os processos – transmissão de mensagens, salvaguarda de dados, ou execução de software – são seguros e confiáveis. O enunciado da segunda fase será oportunamente divulgado.

O trabalho consiste na concretização de um sistema simplificado de armazenamento de ficheiros, designado por ***mySharing***, onde o utilizador recorre a um servidor central para armazenar e, se desejado, partilhar os seus ficheiros. A arquitetura cliente-servidor que se pretende implementar é representada na Figura 1.

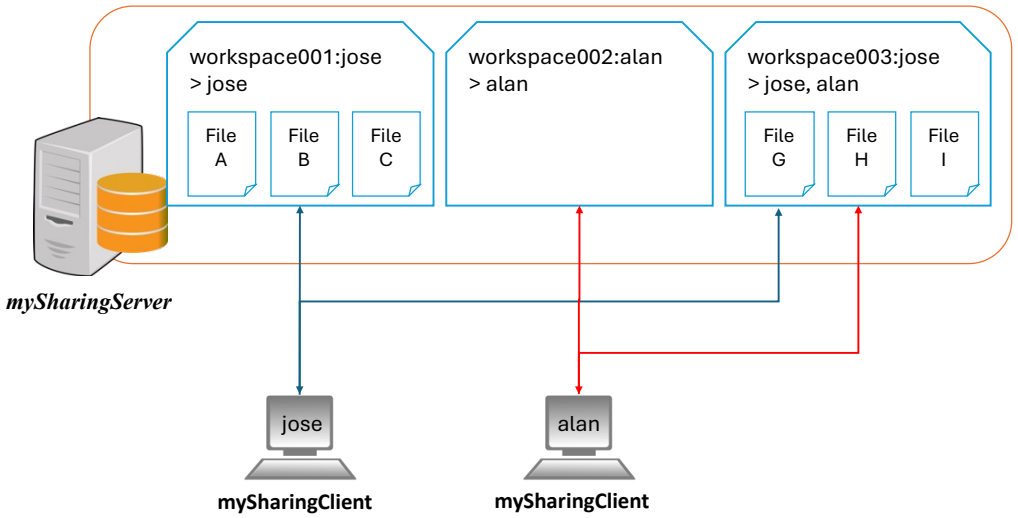

*Figura 1 Diagrama da arquitetura do sistema **mySharing**, com um servidor e múltiplos clientes.*

O cliente ***mySharingClient*** é responsável por enviar e receber ficheiros para/do servidor. Este programa é executado em cada máquina cliente e requer a identificação de um utilizador único (por exemplo, o utilizador `alan`). Para um cliente enviar, receber ou apagar ficheiros do servidor, o utilizador deverá autenticar-se previamente no servidor.

O servidor ***mySharingServer*** implementa o conceito de `workspace`, onde cada `workspace` pode pertencer a um único utilizador (exemplo do `workspace001` e `workspace002` representado na Figura 1) ou pode ser partilhado por vários utilizadores (exemplo do `workspace003` representado na Figura 1). Os utilizadores

podem ter acesso a múltiplos workspaces. Desta forma, um utilizador com acesso a um dado workspace pode aceder, alterar ou remover os ficheiros lá existentes. Cada `workspace` tem um, e apenas um, proprietário (`Owner`). Apenas o `Owner` pode adicionar outros utilizadores a esse `workspace`. Na Figura 1, o `Owner` de cada workspace aparece a seguir ao nome do workspace (e.g., `workspace001:jose`), e ele é também membro do workspace (`> jose, alan`).

A aplicação ***mySharingClient*** têm como objetivo permitir a interação do utilizador com o servidor ***mySharingServer***. Ela deve permitir a criação de workspaces, a adição de utilizadores aos workspaces, o envio de ficheiros (upload), a receção de ficheiros (download) e a remoção de ficheiros no servidor. Para fazer uso de todas estas funcionalidades, a aplicação ***mySharingClient*** deverá implementar uma linha de comandos simples, para interação com o utilizador.

A aplicação servidor ***mySharingServer*****,** é um programa que permite a ligação com vários clientes em simultâneo, mantém informação sobre os ficheiros, workspaces e utilizadores registados, autentica e identifica utilizadores, e permite a coleção e partilha de ficheiros pelos vários clientes.

2 Arquitetura do Sistema

O trabalho consiste no desenvolvimento de dois programas:

- A aplicação servidor ***mySharingServer*** (espera ligações TCP)
- A aplicação cliente ***mySharingClient*** (acede ao servidor via TCP)

O servidor irá executar numa máquina e permitir ligar um número alargado de clientes a serem executados simultaneamente em máquinas distintas. A execução das aplicações deve ser feita da seguinte forma:

1. Servidor:

   `mySharingServer[port]`

   - `[port]` identifica o porto (TCP) para aceitar ligações de clientes. Por omissão, o servidor deve usar o porto 12345 e aceitar ligações em qualquer interface de rede.

2. Cliente:

   `mySharingClient <serverAddress> <user-id> <password>`

   Em que:

   - `<serverAddress>` identifica o servidor. O formato de `serverAddress` é o seguinte: `<IP/hostname>[:Port]`. O endereço IP/hostname do servidor é obrigatório e o porto é opcional. Por omissão, o cliente deve ligar-se ao porto 12345 do servidor.
   - `<user-id>` - string que identifica o utilizador no servidor ***mySharingServer***.
   - `<password>` - string correspondente à password do utilizador.

3 Funcionalidades

3.1 Cliente

A aplicação cliente deve ter o seguinte comportamento sequencial:

1. Começa por enviar o *user-id* e a senha ao servidor, para tentar autenticar o utilizador.
2. Um de três cenários podem ocorrer:

a. O cliente não consegue autenticar no servidor – este receberá uma mensagem do tipo `WRONG-PWD`. O utilizador terá oportunidade de voltar a tentar outra password.
b. O cliente não existe registado no servidor – este será registado no servidor, criado um novo `workspace` para o utilizador, e será devolvida uma mensagem do tipo `OK-NEW-USER` para o cliente.
c. O cliente existe e a password está correta – o cliente recebe uma mensagem do tipo `OK-USER`.

3. Em seguida, a aplicação ***mySharingClient*** deverá apresentar um menu com os comandos disponíveis ao utilizador, nomeadamente o seguinte texto:

- **`CREATE <ws>`** `# Criar um novo workspace - utilizador é Owner.`

- **`ADD <user1> <ws>`** `# Adicionar utilizador` **`<user1>`** `ao workspace` **`<ws>.`**
  `A operação` **`ADD`** `só funciona se o utilizador for o Owner do workspace` **`<ws>.`**

- **`UP <ws> <file1> ... <filen>`** `# Adicionar ficheiros ao workspace.`
- **`DW <ws> <file1> ... <filen>`** `# Download de ficheiros do workspace para a máquina local.`
- **`RM <ws> <file1> ... <filen>`** `# Apagar ficheiros do workspace.`
- **`LW`** `# Lista os workspaces associados ao utilizador.`
- **`LS <ws>`** `# Lista os ficheiros dentro de um workspace.`

  Nas operações **`UP, DW`** e **`RM,`** o número de ficheiros não é limitado, i.e., o utilizador pode indicar múltiplos ficheiros de uma só vez. As operações **`UP, DW, RM`** e **`LS`** só serão válidas se o utilizador fizer parte do grupo de utilizadores do workspace **`<ws>.`**

4. A aplicação cliente deverá implementar uma linha de comandos que permita a utilização dos comandos apresentados no passo 3.
5. O utilizador introduz um dos comandos disponíveis, e um pedido respetivo é enviado ao servidor. O cliente recebe a resposta do servidor e imprime o resultado da operação no terminal.
6. Enquanto o utilizador não pressionar CTRL+C, o programa deverá voltar ao passo 3. Caso contrário, termina.

Comandos Disponíveis

`O comando CREATE <ws>` deverá criar um novo `workspace` no servidor. Caso o servidor aceite o pedido, o cliente deve receber uma mensagem do tipo `OK`. Caso o `workspace` já exista, deverá receber uma mensagem do tipo `NOK`.

`Exemplo`

`Comando:` **`CREATE workspace004`**

`Resposta: OK`

O comando **`ADD <user1> <ws>`** deve permitir adicionar o utilizador `<user1>` ao workspace `<ws>`. Caso o servidor aceite o pedido, o cliente deverá receber uma mensagem do tipo `OK`.

Se o utilizador que faz o pedido não é o `Owner` do `workspace`, o `workspace` não existe, ou o utilizador não

existe, então o cliente deverá receber uma mensagem do tipo NOPERM, NOWS, ou NOUSER, respetivamente.

Exemplo:

Utilizador: **jose**

Comando: **ADD alan workspace003**

Resposta: OK

Comando: **ADD alan workspace004**

Resposta: NOWS # esse workspace não existe

Utilizador: **alan**

Comando: **ADD alan workspace003**

Resposta: NOPERM # sem permissões

O comando **UP <ws> <file1> ... <filen>** permite adicionar os ficheiros <file1> ... <filen> ao workspace <ws>.

Caso algum dos ficheiros indicados não exista no cliente, este deve indicar que o ficheiro não existe e prosseguir para o próximo ficheiro na lista, ou terminar caso não existam mais ficheiros para enviar.

Para cada ficheiro recebido pelo servidor, o cliente deve receber uma mensagem do tipo OK. Caso o utilizador atual não pertença ao workspace <ws>, ou o workspace não exista o cliente deverá receber uma mensagem do tipo NOPERM ou NOWS, respetivamente.

Exemplo:

Utilizador: **jose**

Ficheiros existentes no cliente: **file1.txt file3.jpg**

(file2.pdf não existe no cliente)

Comando: **UP workspace003 file1.txt file2.pdf file3.jpg**

Resposta: file1.txt: OK
file2.pdf: Não Existe
file3.jpg: OK

Comando: **UP workspace004 file1.txt file2.pdf file3.jpg**

Resposta: NOWS # esse workspace não existe

Comando: **UP workspace002 file1.txt file2.pdf file3.jpg**

Resposta: NOPERM # sem permissões

O comando **DW <ws> <file1> ... <filen>** permite fazer download dos ficheiros <file1> ... <filen> existentes no workspace <ws> para a máquina cliente.

Caso o utilizador atual não pertença ao workspace <ws>, ou o workspace não exista o cliente deverá receber uma mensagem do tipo NOPERM ou NOWS, respetivamente.

Caso algum dos ficheiros indicados não exista no workspace, o servidor deve indicar que o ficheiro não existe e prosseguir para o próximo ficheiro na lista, ou terminar caso não existam mais ficheiros para enviar.

Exemplo:
Utilizador: **jose**
Ficheiros existentes no workspace003: **file1.txt file3.jpg**
(file2.pdf não existe no workspace003)

Comando: **DW workspace003 file1.txt file2.pdf file3.jpg**
Resposta: file1.txt #ficheiro transferido
O ficheiro file2.pdf não existe no workspace indicado
file3.jpg #ficheiro transferido

Comando: **DW workspace004 file1.txt file2.pdf file3.jpg**
Resposta: NOWS # esse workspace não existe

Comando: **DW workspace002 file1.txt file2.pdf file3.jpg**
Resposta: NOPERM # sem permissões

O comando **RM <ws> <file1> ... <filen>** permite apagar os ficheiros <file1> ... <filen> existentes no workspace <ws>.

Caso o utilizador atual não pertença ao workspace <ws>, ou o workspace não exista o cliente deverá receber uma mensagem do tipo NOPERM ou NOWS, respetivamente.

Caso algum dos ficheiros indicados não exista no workspace, o servidor deve indicar que o ficheiro não existe e prosseguir para o próximo ficheiro na lista, ou terminar caso não existam mais ficheiros para remover.

Exemplo:
Utilizador: **jose**
Ficheiros existentes no workspace003: **file1.txt file3.jpg**
(file2.pdf não existe no workspace003)

Comando: **RM workspace003 file1.txt file2.pdf file3.jpg**
Resposta: file1.txt: APAGADO
O ficheiro file2.pdf não existe no workspace indicado
file3.jpg: APAGADO

Comando: **RM workspace004 file1.txt file2.pdf file3.jpg**

Resposta: NOWS # esse workspace não existe

Comando: **RM workspace002 file1.txt file2.pdf file3.jpg**

Resposta: NOPERM # sem permissões

O comando **LW** lista os workspaces associados ao utilizador. Uma mensagem com a lista de nomes de workspaces deve ser gerada pelo servidor e enviada ao cliente.

Note que o workspace associado ao próprio utilizador é criado no seu registo, pelo que esta opção irá devolver, pelo menos, o nome desse workspace.

Exemplo

Utilizador: **jose**

Comando: **LW**

Resposta: { **workspace001, workspace003** }

O comando **LS <ws>** lista os ficheiros existentes no workspace <ws>. Caso o utilizador atual não pertença ao workspace <ws>, ou o workspace não exista o cliente deverá receber uma mensagem do tipo NOPERM ou NOWS, respetivamente. Caso contrário, uma mensagem com a lista de ficheiros do <ws> deve ser gerada e enviada ao cliente.

Exemplo

Utilizador: **jose**

Ficheiros no workspace003: **file1.txt file3.jpg**

Comando: **LS workspace003**

Resposta: **{ file1.txt, file3.jpg }**

Comando: **LS workspace004**

Resposta: NOWS # esse workspace não existe

Comando: **LS workspace002**

Resposta: NOPERM # sem permissões

3.2 Servidor

O servidor mantém um **ficheiro de texto** com os **utilizadores** do sistema e as suas respetivas passwords. Este ficheiro deverá ser utilizado quando o cliente se tenta autenticar com o servidor.

O servidor mantém também um **ficheiro de texto** com os **workspaces** existentes, nomeadamente o seu nome, o

`Owner` e os utilizadores que lhe estão associados. Este ficheiro deverá ser utilizado sempre que um cliente tenta fazer qualquer operação sobre um workspace.

Caso um utilizador se tente autenticar, e não esteja registado no servidor, o Servidor fará o registo do novo utilizador, adicionando o *user-id* e a respetiva *senha* ao ficheiro de utilizadores do servidor. Devem também ser inicializadas as estruturas de dados que mantêm informações relativas a cada utilizador.

4 Entrega

O prazo de entrega é no dia **28 de março, até às 23:59 horas**.

O código desta primeira fase do trabalho deve ser entregue de acordo com as seguintes regras:

- Para entregar o trabalho, é necessário criar um **ficheiro zip** com o nome **SegC-grupoXX-proj1-fase1.zip**, onde **XX** é o número do grupo, contendo:
  - ✓ o código do trabalho;
  - ✓ os ficheiros jar (cliente e servidor) para execução do projeto;
  - ✓ um readme (txt) indicando **como compilar** e **como executar** o projeto, indicando também as limitações do trabalho.
- O ficheiro zip é submetido através da atividade disponibilizada para esse efeito na página da disciplina no Moodle. Apenas um dos elementos do grupo deve submeter. Se existirem várias submissões do mesmo grupo, será considerada a mais recente.
- Não serão aceites trabalhos enviados por email. Se não se verificar algum destes requisitos o trabalho é considerado não entregue.

5 Autoavaliação de contribuições

Cada aluno tem de preencher no Moodle um formulário de autoavaliação das contribuições individuais de cada elemento do grupo para o projeto. Por exemplo, se todos os elementos colaboraram de forma idêntica, bastará que todos indiquem que cada um contribuiu 33%. Aplicam-se as seguintes regras e penalizações:

- Alunos que não preencham o formulário **sofrem uma penalização na nota de 10%.**
- Caso existam assimetrias significativas entre as respostas de cada elemento do grupo, o grupo poderá ser chamado para as explicar.
- Se as contribuições individuais forem diferentes, isso será refletido na nota de cada elemento do grupo, levando à atribuição de notas individuais diferentes.

O prazo de preenchimento é o mesmo que o da entrega do projeto (**28 de março, até às 23:59 horas**).